1.3. A complete example: The *readme* DTD

The reason for *readme* was that I often wrote two versions of files such as README and INSTALL which explain aspects of a distributed software archive; one version was ASCII-formatted, the other was written in HTML. Maintaining both versions means double amount of work, and changes of one version may be forgotten in the other version. To improve this situation I invented the *readme* DTD which allows me to maintain only one source written as XML document, and to generate the ASCII and the HTML version from it.

In this section, I explain only the DTD. The *readme* DTD is contained in the **Markup** distribution together with the two converters to produce ASCII and HTML. Another section of this manual describes the HTML converter.

The documents have a simple structure: There are up to three levels of nested sections, paragraphs, item lists, footnotes, hyperlinks, and text emphasis. The outermost element has usually the type `readme`, it is declared by

```
<!ELEMENT readme (sect1+)>
<!ATTLIST readme
          title CDATA #REQUIRED>
```

This means that this element contains one or more sections of the first level (element type `sect1`), and that the element has a required attribute `title` containing character data (CDATA). Note that `readme` elements must not contain text data.

The three levels of sections are declared as follows:

```
<!ELEMENT sect1 (title,(sect2|p|ul)+)>

<!ELEMENT sect2 (title,(sect3|p|ul)+)>

<!ELEMENT sect3 (title,(p|ul)+)>
```

Every section has a `title` element as first subelement. After the title an arbitrary but non-empty sequence of inner sections, paragraphs and item lists follows. Note that the inner sections must belong to the next higher section level; `sect3` elements must not contain inner sections because there is no next higher level.

Obviously, all three declarations allow paragraphs (`p`) and item lists (`ul`). The definition can be simplified at this point by using a parameter entity:

```
<!ENTITY % p.like "p|ul">

<!ELEMENT sect1 (title,(sect2|%p.like;)+)>

<!ELEMENT sect2 (title,(sect3|%p.like;)+)>

<!ELEMENT sect3 (title,(%p.like;)+)>
```

Here, the entity `p.like` is nothing but a macro abbreviating the same sequence of declarations; if new

elements on the same level as `p` and `ul` are later added, it is sufficient only to change the entity definition. Note that there are some restrictions on the usage of entities in this context; most important, entities containing a left paranthesis must also contain the corresponding right paranthesis.

Note that the entity `p.like` is a *parameter* entity, i.e. the ENTITY declaration contains a percent sign, and the entity is referred to by `%p.like;`. This kind of entity must be used to abbreviate parts of the DTD; the *general* entities declared without percent sign and referred to as `&name;` are not allowed in this context.

The `title` element specifies the title of the section in which it occurs. The title is given as character data, optionally interspersed with line breaks (`br`):

```
<!ELEMENT title (#PCDATA|br)*>
```

Compared with the `title` *attribute* of the `readme` element, this element allows inner markup (i.e. `br`) while attribute values do not: It is an error if an attribute value contains the left angle bracket < literally such that it is impossible to include inner elements.

The paragraph element `p` has a structure similar to `title`, but it allows more inner elements:

```
<!ENTITY % text "br|code|em|footnote|a">

<!ELEMENT p (#PCDATA|%text;)*>
```

Line breaks do not have inner structure, so they are declared as being empty:

```
<!ELEMENT br EMPTY>
```

This means that really nothing is allowed within `br`; you must always write `<br></br>` or abbreviated `<br/>`.

Code samples should be marked up by the `code` tag; emphasized text can be indicated by `em`:

```
<!ELEMENT code (#PCDATA)>

<!ELEMENT em (#PCDATA|%text;)*>
```

That `code` elements are not allowed to contain further markup while `em` elements do is a design decision by the author of the DTD.

Unordered lists simply consists of one or more list items, and a list item may contain paragraph-level material:

```
<!ELEMENT ul (li+)>

<!ELEMENT li (%p.like;)*>
```

Footnotes are described by the text of the note; this text may contain text-level markup. There is no mechanism to describe the numbering scheme of footnotes, or to specify how footnote references are printed.

```
<!ELEMENT footnote (#PCDATA|%text;)*>
```

Hyperlinks are written as in HTML. The anchor tag contains the text describing where the link points to, and the `href` attribute is the pointer (as URL). There is no way to describe locations of "hash marks". If

the link refers to another *readme* document, the attribute `readmeref` should be used instead of `href`. The reason is that the converted document has usually a different system identifier (file name), and the link to a converted document must be converted, too.

```
<!ELEMENT a (#PCDATA)*>
<!ATTLIST a
          href      CDATA #IMPLIED
          readmeref CDATA #IMPLIED
>
```

Note that although it is only sensible to specify one of the two attributes, the DTD has no means to express this restriction.

So far the DTD. Finally, here is a document for it:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE readme SYSTEM "readme.dtd">
<readme title="How to use the readme converters">
<sect1>
  <title>Usage</title>
  <p>
    The <em>readme</em> converter is invoked on the command line by:
  </p>
  <p>
    <code>readme [ -text | -html ] input.xml</code>
  </p>
  <p>
    Here a list of options:
  </p>
  <ul>
    <li>
      <p><code>-text</code>: specifies that ASCII output should be produced</p>
    </li>
    <li>
      <p><code>-html</code>: specifies that HTML output should be produced</p>
    </li>
  </ul>
  <p>
    The input file must be given on the command line. The converted output is
    printed to <em>stdout</em>.
  </p>
</sect1>
<sect1>
  <title>Author</title>
  <p>
    The program has been written by
    <a href="mailto:Gerd.Stolpmann@darmstadt.netsurf.de">Gerd Stolpmann</a>.
  </p>
</sect1>
</readme>
```